

Context Under Pressure: How Language Model Agents Should Save and Read Information Over Long Interactions

Stanford CS224N Custom Project

Alexander Worley
Department of Computer Science
Stanford University
aworley@stanford.edu

Abstract

Language model agents that interact over many turns must decide what past information to keep in the prompt and what to discard. This decision becomes critical under a fixed token budget, where including irrelevant history can harm performance just as much as losing important facts. I present a controlled empirical study of context management strategies for LLM agents across two task families, multi-topic fact recall and constraint satisfaction, designed to stress memory in fundamentally different ways. I implement and evaluate eleven strategies, ranging from simple baselines like no memory and sliding window, through LLM-augmented approaches like global summary and key-fact extraction, to a novel structured constraint ledger that uses LLM-based extraction to organize past information into typed fields. I also introduce two task-aware meta-strategies that route conversations to the best-fit strategy. Experiments using Gemini 2.0 Flash show that no single fixed strategy dominates both task families. The meta-prompted routing strategy achieves the best overall performance, reaching 83.3% exact match on recall and 75.2% constraint satisfaction, surpassing even an oracle upper bound on constraint satisfaction at 73.3%. An ablation study reveals that LLM-based extraction matters far more than label formatting, and noise injection experiments demonstrate graceful degradation under up to 30% irrelevant content.

1 Key Information

- Mentor: David Anugraha
- External Collaborators: No
- Sharing project: No
- Late Days: 1

2 Introduction

Large language models are typically evaluated on short tasks where all relevant information fits within a single prompt. In realistic deployments, however, interactions can span dozens of turns. Instructions accumulate gradually, topics shift and return, and important facts stated early may only become relevant much later. In these settings, the model’s context window becomes a scarce resource that must be managed carefully. This resource management problem is not solved by simply increasing context length. Even models with large context windows do not always use long inputs effectively. Relevant information can be buried among irrelevant turns, a phenomenon studied as the “lost in the middle” effect [1]. What matters is not just *what* information is available, but *how* it is organized in the prompt.

Park et al. [2] introduced Generative Agents, demonstrating that explicit memory systems where an agent writes, retrieves, and uses memories can produce coherent long-term behavior in simulated environments. However, their work leaves open several empirical questions. They do not isolate which components of their memory system drive the observed improvements, do not test under fixed token budgets where strategies must compete for the same prompt space, and do not stress-test against noisy histories. I address these gaps with a controlled study of context management strategies under fixed budgets.

First, I build an evaluation framework with two task families that stress memory in complementary ways, multi-topic fact recall and constraint satisfaction, both using synthetic data with deterministic evaluation. Second, I implement eleven context management strategies, including a novel structured constraint ledger that uses LLM-based extraction to organize conversation history into typed fields, and two task-aware meta-strategies that route conversations to the best-fit strategy. Third, I run comprehensive experiments covering all strategies across three token budgets and three noise levels, evaluated on 50 conversations per strategy per family using Gemini 2.0 Flash [3]. My main findings are that no single fixed strategy wins both families, that task-aware routing achieves the best overall performance, that LLM-based extraction matters far more than label formatting, and that all strategies degrade gracefully under noise.

3 Related Work

The idea that memory is central to language model agents was established by Park et al. [2], whose Generative Agents architecture combines episodic storage, retrieval by relevance and recency, and periodic reflection. Their evaluation was primarily qualitative, conducted in a simulated town environment, and did not isolate which components of the memory system mattered most. My work complements theirs by providing controlled quantitative evaluation under fixed budgets, with automatic metrics that measure exactly when and why each strategy succeeds or fails. A related challenge is that even models with large context windows do not always use long inputs effectively. Liu et al. [1] showed that language models struggle to retrieve information from the middle of long contexts, performing best when relevant information appears at the beginning or end. This “lost in the middle” effect motivates my approach of actively managing what appears in the prompt rather than relying on the model to attend to the right information in a long history.

My work also connects to retrieval-augmented generation, introduced by Lewis et al. [4], which combines a retrieval module with a generative model to fetch relevant documents at inference time. My strategies share the same high-level structure of retrieving relevant information and then generating, but operate over conversation history that accumulates and changes across turns rather than a static document store. Yao et al. [5] proposed ReAct, interleaving reasoning and acting in language model agents, focusing on deciding what actions to take. My setting addresses the complementary problem of managing information the agent has already received. Finally, Chevalier et al. [6] proposed methods to compress long contexts into shorter representations. My global summary and key-fact strategies implement similar ideas at the prompt level, and my results show that the format of compression matters as much as the compression ratio.

4 Approach

4.1 Problem Formulation

Consider a conversation with history $h = (u_1, a_1, u_2, a_2, \dots, u_t)$ where u_i and a_i are user and assistant turns respectively. At each decision point, a context strategy f must construct a context string $c = f(h, B)$ of at most B tokens, which is concatenated with the current query and sent to the language model. The strategy maintains an internal memory state M that is updated after each observed turn via $M_t = \text{update}(M_{t-1}, u_t)$. I evaluate two objectives. The first is **downstream task performance**, meaning the quality of the model’s responses when conditioned on the constructed context. The second is **context retrieval rate**, meaning the fraction of gold items that appear in the constructed context string c . This second metric separates the question of whether the strategy retrieves the right information from whether the model uses that information correctly. As I show in my error analysis, this distinction turns out to be important, since almost all failures occur when the

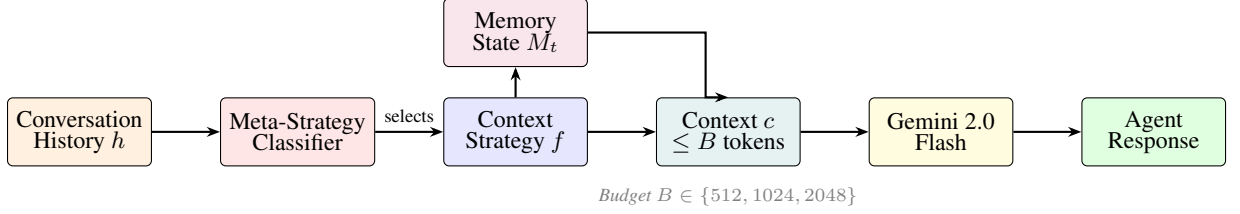


Figure 1: System pipeline. The meta-strategy classifies the conversation and selects a context strategy. The strategy maintains a memory state M_t updated at each turn, serializes it into a context string c within the token budget, and passes it to the LLM for generation.

gold information *is* present in the context but the model fails to use it. Figure 1 illustrates the overall pipeline.

4.2 Task Families

Family A is multi-topic fact recall. Each conversation spans 3 to 6 topics such as pets, food, and fitness. The user states 3 to 5 gold facts per topic across many turns, with distractor facts interspersed. At 6 designated recall points, the user asks the agent to retrieve a specific fact from an earlier topic. **Family B** is constraint satisfaction. Each conversation introduces 10 to 25 constraints gradually across turns in one of 6 domains such as study plans, workout routines, and meal plans. Constraints include required items, forbidden items, time limits, and formatting rules. At a final prompt, the agent must produce an output satisfying all constraints. These families are complementary. Family A rewards precise retrieval of specific facts from the correct context, while Family B rewards comprehensive retention of all accumulated requirements.

4.3 Strategies

I implement eleven strategies organized into four categories.

Baselines. *No memory* provides the model with only the current query and no history, serving as a lower bound. *Sliding window* includes the most recent turns that fit within the budget.

LLM-augmented strategies. These strategies use the language model itself to process history at each turn. *Global summary* maintains a running summary of the conversation, periodically re-summarizing when the summary grows too long. *Key-fact extraction* prompts the model to extract individual facts from each user turn into a flat list. *Topic bucket* maintains separate memory stores per topic, routing each user turn to the appropriate bucket via prompted topic classification.

Structured constraint ledger. This is my primary contribution. The strategy prompts the LLM at each user turn to extract information into a typed ledger with labels such as REQUIRED, FORBIDDEN, FORMAT, and PREFERENCE. At decision time, the full ledger is injected into the prompt. The design hypothesis is that organizing information by type rather than by time or topic helps the model attend to all relevant constraints simultaneously. Formally, after observing user turn u_t , the strategy updates its memory as

$$M_t = M_{t-1} \cup \text{LLM}_{\text{extract}}(u_t, \text{labels}) \quad (1)$$

where $\text{LLM}_{\text{extract}}$ is a prompted extraction that returns typed key-value pairs. The context at decision time is $c = \text{serialize}(M_t)$, truncated to B tokens. This is not a prompt-only approach. The LLM performs genuine extraction and reformulation at each observed turn, building a persistent data structure across the conversation. Table 1 shows a real example from my experiments where 13 constraints from a 40-turn meal-planning conversation are compressed into a structured ledger. I test two ablations to isolate which components matter. *SC regex* replaces LLM extraction with keyword-based pattern matching. *SC no label* uses LLM extraction but omits the type labels.

Oracle upper bound. The oracle injects the gold facts or gold constraints directly into the prompt, representing perfect information retrieval. This is not a deployable strategy but establishes a ceiling on task performance.

Table 1: Real example of structured constraint extraction from conversation B-0001 in the test set. Thirteen constraints spread across 40 turns are compressed into an organized ledger.

Raw user turns	Extracted ledger
“Avoid anchovies” “No eggplant please” “Avoid olives” “Leave out mushrooms”	MUST NOT INCLUDE: anchovies, eggplant, olives, mushrooms
“Include salmon at least once” “I’d love quinoa at some point”	MUST INCLUDE: sweet potatoes, salmon, quinoa
“Plan for 2 meals per day” “No more than 400 cal per meal”	NUMERIC: 2 meals/day, 400 cal, \$75 budget, 45 min
“Present it as a table”	FORMAT: table, per day

Meta-strategies. I implement two task-aware routing strategies that identify whether a conversation is closer to Family A or Family B and delegate to the best-fit strategy. For recall-heavy conversations, the meta-strategy delegates to sliding window. For constraint-heavy conversations, it delegates to structured constraint. *Meta-prompted* asks the LLM to classify the conversation type before each decision, adding one LLM call per decision point. *Meta-learned* trains a logistic regression classifier on the full development set of 160 conversations using five surface features: the fraction of turns containing constraint language, the count of topic-related keywords, whether the final turn requests a plan, the average turn length, and the total turn count. This requires no LLM calls at routing time.

5 Experiments

5.1 Data

I generate 500 Family A conversations and 300 Family B conversations using template-based generation with randomized topics, facts, constraints, and distractor content. Family A conversations average 79 turns with 6 recall questions each. Family B conversations average 54 turns with a single final prompt. Data is split into 20% development and 80% test. The development set was used exclusively for prompt tuning and hyperparameter selection. All reported results are on the held-out test set. Prompt templates for each domain are human-written seeds. Distractor facts for Family A are drawn from topics not present in the conversation to ensure surface-level similarity without semantic overlap. I validate that no duplicate fact sets or constraint sets exist using automated checks, and I manually inspected 50 conversations per family to verify that gold answers are correct and unambiguous.

5.2 Evaluation Method

For Family A, I measure **exact match** after lowercasing and whitespace cleanup, **token F1** for token-level overlap, and **contradiction rate** for answers conflicting with gold facts. For Family B, I measure **satisfaction rate** as the average fraction of constraints satisfied using a deterministic checker, **strict success rate** as the fraction of conversations with all constraints met, and **average violation count**. For both families, I measure **context retrieval rate**, the fraction of gold items appearing in the constructed context at decision time, which isolates retrieval quality from generation quality.

5.3 Experimental Details

All experiments use Gemini 2.0 Flash [3] via Vertex AI at temperature 0.3. Token counting uses an approximate method of 1 token per 4 characters. Family A responses are capped at 512 output tokens and Family B at 2048. Rate limiting is handled by exponential backoff starting at 10 seconds. Each experiment runs 50 conversations per family, for a total of 45 experimental configurations.

Table 2: Main comparison at $B = 1024$. All values are percentages except B Viol, which is an average count.

Strategy	Family A				Family B			
	EM	F1	Con	Ret	Sat	Str	Viol	Ret
No Memory	0.4	4.0	0.3	0.0	43.6	0	6.65	0.0
SC Regex	8.4	10.4	1.5	24.3	66.8	3	3.74	100.0
Topic Bucket	69.2	82.0	0.2	91.0	63.1	7	4.20	99.9
Global Summary	82.1	91.6	0.2	97.6	65.0	4	4.01	100.0
Sliding Window	82.6	92.3	0.5	98.5	64.6	9	4.03	100.0
Key Fact	73.5	86.6	0.4	96.4	70.3	13	3.34	100.0
Struct. Constraint	78.0	88.4	0.5	97.0	70.0	18	3.28	100.0
SC No Label	80.6	90.7	0.5	97.3	71.6	11	3.26	100.0
Meta-Learned	83.1	92.4	0.5	98.5	72.9	16	3.07	100.0
Oracle	93.0	97.0	0.2	100.0	73.3	15	3.02	100.0
Meta-Prompted	83.3	92.2	0.5	98.5	75.2	20	2.77	100.0

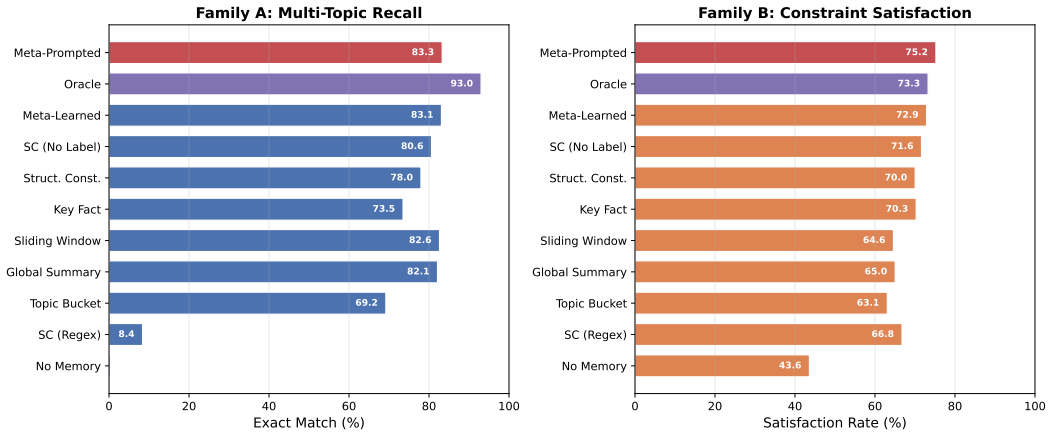


Figure 2: Strategy comparison at $B = 1024$. No single strategy wins both families. Meta-Prompted is the only non-oracle strategy to achieve strong performance on both.

5.4 Results

Table 2 shows the main comparison at budget $B = 1024$, and Figure 2 visualizes the results. Four key findings emerge. First, no single fixed strategy wins both families. Sliding window achieves strong recall at 82.6% EM but weak constraint satisfaction at 64.6%, while structured constraint shows the opposite trade-off with 78.0% recall and 70.0% satisfaction. This confirms the hypothesis that different task structures require different memory behaviors. Second, task-aware routing is the best overall approach. Meta-prompted reaches 83.3% EM on Family A and 75.2% satisfaction on Family B, the best non-oracle result on both families and the only strategy to surpass the oracle on Family B. Third, prompt format matters as much as content. Meta-prompted surpasses the oracle on Family B at 75.2% versus 73.3% because it organizes information into typed categories rather than presenting raw text. Even with perfect information, an unstructured presentation leaves performance on the table. Fourth, context retrieval rate reveals a “last mile” problem. Most LLM-augmented strategies retrieve above 96% of gold items, yet downstream accuracy is substantially lower, meaning the bottleneck is using information correctly rather than finding it. My error analysis in Section 6.3 confirms this.

5.4.1 Budget Sweep

Figure 3 shows performance at budgets 512, 1024, and 2048, with full results in Appendix A. At budget 512, all non-oracle strategies degrade significantly. Sliding window drops to 37.8% EM on Family A because the window only fits a few recent turns. Structured constraint maintains stronger Family B performance at 73.3% because its compressed ledger fits more constraints into fewer tokens. At budget 2048, most strategies converge on Family A as windows become large enough to include

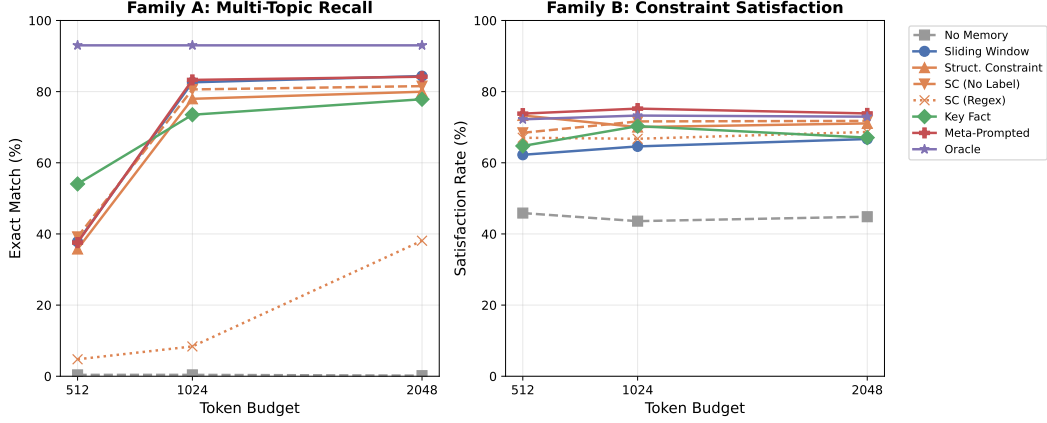


Figure 3: Performance across token budgets. At $B = 512$, sliding window loses earlier topics and degrades sharply on Family A. Structured constraint maintains strong Family B performance because its compressed ledger uses tokens efficiently.

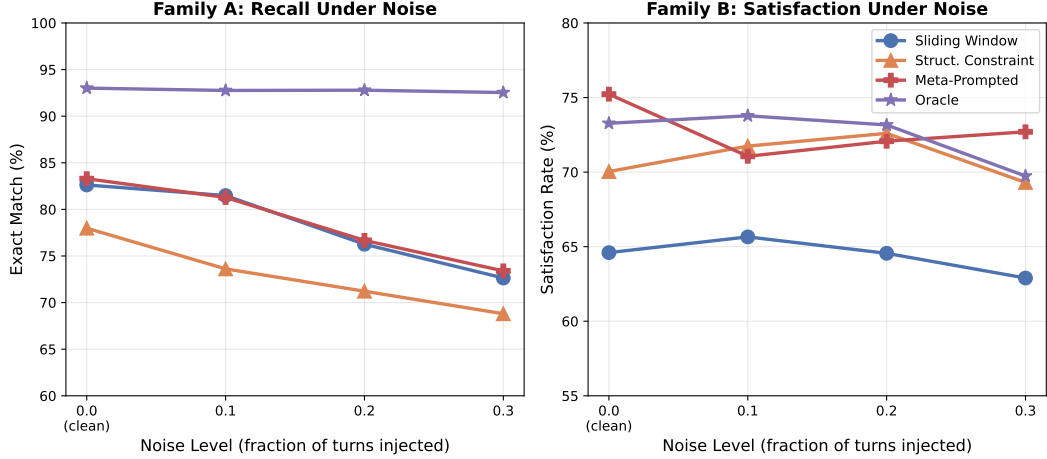


Figure 4: Noise robustness at $B = 1024$. Noise is injected as irrelevant filler turns. Family A degrades gradually. Family B is remarkably stable, especially for structured constraint.

nearly all relevant turns, but Family B shows less convergence, suggesting that constraint satisfaction benefits more from organizational structure than from raw context quantity. Meta-prompted maintains its advantage across all three budgets.

5.4.2 Noise Robustness

Figure 4 shows performance under noise injection at 10%, 20%, and 30%, with full results in Appendix B. Noise is injected by inserting irrelevant filler turn pairs at random positions. Family A recall degrades gradually, with oracle nearly unaffected at 93.0% to 92.5% and structured constraint showing the smallest drop among deployable strategies at 9.2 points compared to 10.0 for sliding window. The extraction step acts as a noise filter, discarding filler content. Family B satisfaction is remarkably stable, with structured constraint dropping only 0.7 points from clean to 30% noise because the extraction prompt naturally ignores irrelevant filler turns.

Table 3: Ablation of structured constraint components at $B = 1024$.

Variant	Extraction	A EM	B Sat	B Viol
SC full	LLM	78.0	70.0	3.28
SC no label	LLM	80.6	71.6	3.26
SC regex	Pattern match	8.4	66.8	3.74

6 Analysis

6.1 Ablation: What Makes the Structured Ledger Work?

Table 3 isolates the contributions of LLM extraction and type labels. The regex variant, which uses keyword pattern matching instead of LLM extraction, collapses to 8.4% on Family A, barely above the no-memory baseline. This demonstrates that the LLM’s ability to understand and extract relevant information is the key ingredient, not the ledger format itself, and directly addresses the milestone feedback asking whether structured constraint is “prompt-based only.” The no-label variant actually outperforms the full labeled version on both families, reaching 80.6% versus 78.0% on Family A and 71.6% versus 70.0% on Family B. One possible explanation is that the classification step sometimes misclassifies information. Another is that labels consume tokens that could hold additional extracted content.

6.2 Meta-Strategy Comparison

Both meta-strategies outperform every fixed strategy, confirming that task-aware routing adds value regardless of the routing mechanism. Meta-prompted slightly outperforms meta-learned on Family B at 75.2% versus 72.9%, at the cost of one additional LLM call per decision point. The close performance of the logistic regression classifier suggests that the classification task is largely learnable from surface features, though the LLM captures some additional signal. The success of meta-learned is notable because it shows routing does not require an expensive LLM call, making the approach practical in latency-sensitive settings.

6.3 Error Analysis

I examined all failures from meta-prompted at budget 1024. Family A had 12 genuine failures out of 431 questions after normalizing for trivial differences like pronoun shifts. In 8 of 12 failures, the gold answer was present in the context but the model gave the wrong fact, for example answering “The Training Ground” when “Basecamp Fitness” was in the context. In 3 of 12, the gold answer was present but the model refused to answer. Only 1 failure was a true retrieval failure. The average turn gap for failures was 50 turns compared to roughly 40 overall, suggesting longer gaps correlate with more errors.

Family B had 223 violations out of 923 checkable constraints across 50 conversations. Of these, 105 were required items missing from the output, 95 were forbidden items incorrectly included, and 23 were dietary restrictions missed. All 223 violations occurred when the constraint was present in the context, confirming what I call the “attention bottleneck” hypothesis, where the model has all needed information but fails to attend to every constraint during long generation. The pattern across both families is the same. The dominant failure mode is not retrieval failure but generation failure. This explains why different strategies excel at different families: sliding window puts facts in an easy-to-find position near the end of the context, while structured constraint organizes constraints by type, helping the model track them during generation.

7 Conclusion

I presented a controlled study of context management strategies for language model agents under fixed token budgets. No single strategy wins both task families, confirming that different interaction patterns require different memory behaviors. Task-aware routing via meta-prompted classification achieves the best overall performance and is the only non-oracle strategy to surpass the oracle on constraint satisfaction, demonstrating that how information is organized matters as much as what

information is included. The ablation shows that LLM-based extraction is the critical component, while type labels provide no benefit. All strategies degrade gracefully under noise, with structured approaches showing particular resilience.

My error analysis reveals that the dominant failure mode is generation failure, not retrieval failure. The right information is almost always present in the context, but the model either ignores it or confuses it with a similar fact. This suggests that future work should focus not only on what to put in the context but on how to format it to make the model more likely to use it correctly. Combining context management with generation-time techniques like chain-of-thought constraint checking could further improve performance.

Limitations. My evaluation uses synthetic data, which enables automatic scoring but may not capture the full complexity of natural conversations. I use a single model and do not test generalization across providers. With 50 conversations per experiment, closely-performing strategies may not be statistically distinguishable. The meta-strategies assume conversations fall cleanly into one of two families, while real interactions may blend both patterns.

References

- [1] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.
- [2] Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023.
- [3] Google DeepMind Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2024.
- [4] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [5] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- [6] Alexis Chevalier, Alexander Wettig, Anian Clinciu, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

A Appendix: Complete Budget Sweep Results

Table 4: All strategies at three token budgets. Values are Family A Exact Match / Family B Satisfaction in %.

Strategy	$B = 512$	$B = 1024$	$B = 2048$
No Memory	0.4 / 45.9	0.4 / 43.6	0.2 / 44.8
Sliding Window	37.8 / 62.2	82.6 / 64.6	84.4 / 66.7
Global Summary	31.6 / 61.3	82.1 / 65.0	83.8 / 67.2
Key Fact	54.1 / 64.7	73.5 / 70.3	77.9 / 67.1
Topic Bucket	40.1 / 60.6	69.2 / 63.1	76.5 / 63.9
Struct. Constraint	35.8 / 73.3	78.0 / 70.0	80.0 / 71.1
SC Regex	4.8 / 67.0	8.4 / 66.8	38.1 / 68.7
SC No Label	39.2 / 68.4	80.6 / 71.6	81.5 / 71.8
Oracle	93.0 / 72.2	93.0 / 73.3	93.0 / 73.0
Meta-Prompted	37.6 / 73.8	83.3 / 75.2	84.2 / 73.9
Meta-Learned	37.8 / 73.2	83.1 / 72.9	84.8 / 71.3

B Appendix: Noise Robustness Results

Table 5: Noise robustness at $B = 1024$. Values are Family A EM / Family B Satisfaction in %.

Strategy	Clean	10%	20%	30%
Sliding Window	82.6 / 64.6	81.5 / 65.7	76.3 / 64.6	72.6 / 62.9
Struct. Constraint	78.0 / 70.0	73.6 / 71.7	71.2 / 72.6	68.8 / 69.3
Meta-Prompted	83.3 / 75.2	81.3 / 71.1	76.7 / 72.1	73.4 / 72.7
Oracle	93.0 / 73.3	92.8 / 73.8	92.8 / 73.2	92.5 / 69.7

C Appendix: Data Generation and Validation

Family A uses 10 topic types including pets, food, fitness, travel, hobbies, school, work, family, health, and entertainment, each with 3 to 5 gold facts per topic and 3 to 6 topics per conversation. Distractor facts are drawn from absent topics. Family B uses 6 domain types including study plans, workout routines, meal plans, travel itineraries, event plans, and project schedules, each with 10 to 25 constraints. I inspected 50 conversations per family from the test set and confirmed all gold answers are correct, unambiguous, and satisfiable.

D Appendix: Prompts

The structured constraint extraction prompt is: “Extract all facts, preferences, and constraints. Return one per line. Be exhaustive but concise.” For the labeled variant, the prompt additionally requests classification as REQUIRED, FORBIDDEN, FORMAT, or PREFERENCE. The meta-prompted classification prompt is: “Based on the conversation so far, is this interaction primarily about recalling specific facts the user mentioned earlier, or satisfying accumulated constraints and requirements? Reply with just A or B.”

E Appendix: Special Thanks

I would like to give special thanks to my TA mentor David Anugraha, whose feedback on my milestone report inspired the meta-strategy direction that became the strongest contribution of this project.